HAMILTON, BROOK, SMITH & REYNOLDS, P.C.

UTILITY PATENT APPLICATION TRANSMITTAL *(Only for new nonprovisional applications under 37 C.F.R. 1.53(b))*	Attorney Docket No.	AVI99-01
	First Named Inventor or Application Identifier	William J. Dally
	Express Mail Label No.	EL192647737US

Title of Invention: FABRIC ROUTER WITH FLIT CACHING

APPLICATION ELEMENTS

See MPEP chapter 600 concerning utility patent application contents.

ADDRESS TO:
Assistant Commissioner for Patents
Box Patent Application
Washington, D.C. 20231

1. □ Fee Transmittal Form
(Submit an original, and a duplicate for fee processing)

2. ☒ Specification [Total Pages 23]
(preferred arrangement set forth below)
- Descriptive title of the invention
- Cross References to Related Applications
- Statement Regarding Fed sponsored R & D
- Reference to microfiche Appendix
- Background of the Invention
- Summary of the Invention
- Brief Description of the Drawings
- Detailed Description
- Claim(s)
- Abstract of the Disclosure

3. ☒ Drawing(s) (35 U.S.C. 113) [Total Sheets 7]
_____Formal __X__Informal

4. □ Oath or Declaration/POA [Total Pages ____]

a. □ Newly executed (original or copy)

b. □ Copy from a prior application (37 C.F.R. 1.63(d))
(for continuation/divisional with Box 17 completed)
[NOTE Box 5 below]

i. □ DELETION OF INVENTOR(S)
Signed statement attached deleting inventor(s) named in the prior application, see 37 C.F.R. 1.63(d)(2) and 1.33(b).

5. □ Incorporation By Reference *(useable if Box 4b is checked)*
The entire disclosure of the prior application, from which a copy of the oath or declaration is supplied under Box 4b, is considered as being part of the disclosure of the accompanying application and is hereby incorporated by reference therein.

6. □ Microfiche Computer Program *(Appendix)*

7. □ Nucleotide and/or Amino Acid Sequence Submission (if applicable, all necessary)

a. □ Computer Readable Copy

b. □ Paper Copy (identical to computer copy)

c. □ Statement verifying identity of above copies

ACCOMPANYING APPLICATION PARTS

8. □ Assignment Papers (cover sheet & documents)

9. □ 37 C.F.R. 3.73(b) Statement *(when there is an assignee)* □ Power of Attorney

10. □ English Translation Document *(if applicable)*

11. □ Information Disclosure Statement (IDS)/PTO-1449 □ Copies of IDS Citations

12. □ Preliminary Amendment

13. ☒ Return Receipt Postcard (MPEP 503) (2)
(Should be specifically itemized)

14. □ Small Entity Statement(s) □ Statement filed in prior application, Status still proper and desired

15. □ Certified Copy of Priority Document(s)
(if foreign priority is claimed)

16. □ Other:..

17. If a **CONTINUING APPLICATION,** check appropriate box and supply the requisite information:

□ Continuation □ Divisional □ Continuation-in-part (CIP) of prior application No.:______/__________

Prior application information: Examiner: ________________ Group Art Unit: ______________

18. CORRESPONDENCE ADDRESS

NAME	James M. Smith, Esq. HAMILTON, BROOK, SMITH & REYNOLDS, P.C.				
ADDRESS	Two Militia Drive				
CITY	Lexington	STATE	MA	ZIP CODE	02421-4799
COUNTRY	USA	TELEPHONE	(781) 861-6240	FAX	(781) 861-9540

Signature		Date	5/20/99
Submitted by Typed or Printed Name	James M. Smith, Esq.	Reg. Number	28,043

Rev January 12, 1999

K:\JSMITH\AVI\AVI99-01.PA
JMS74
JMS/cjh
May 20, 1999

Date: 5/21/99 Express Mail Label No. EL192647737US

Inventors: William J. Dally, Philip P. Carvey, P. Allen King, William F. Mann and Larry R. Dennison

Attorney's Docket No.: AVI99-01

FABRIC ROUTER WITH FLIT CACHING

BACKGROUND OF THE INVENTION

An interconnection network consists of a set of nodes connected by channels. Such networks are used to transport data packets between nodes. They are used, for example, in multicomputers, multiprocessors, and network switches and routers. In multicomputers, they carry messages between processing nodes. In multiprocessors, they carry memory requests from processing nodes to memory nodes and responses in the reverse direction. In network switches and routers, they carry packets from input line cards to output line cards. For example, published International application PCT/US98/16762 (WO99/11033), by William J. Dally, Philip P. Carvey, Larry R. Dennison and P. Allen King, and entitled "Router With Virtual Channel Allocation," the entire teachings of which are incorporated herein by reference, describes the use of a three-dimensional torus interconnection network to provide the switching fabric for an internet router.

A key issue in the design of Interconnection networks is the management of the buffer storage in the nodes or fabric routers that make up the interconnection fabric. Many prior routers, including that described in the noted pending PCT patent application, manage these buffers using virtual-channel flow control as described in "Virtual Channel Flow Control," by William J. Dally, IEEE Transactions on Parallel and Distributed Systems, Vol. 3, No. 2, March 1992, pp. 194-205. With this method, the buffer space associated with each channel is partitioned and each partition is associated with a virtual channel. Each packet traversing the channel is assigned to a particular virtual channel and does not compete for buffer space with packets traversing other virtual channels.

Recently routers with hundreds of virtual channels have been constructed to provide isolation between different classes of traffic directed to different destinations. With these routers, the amount of space required to hold the virtual channel buffers becomes an issue. The number of buffers, N, must be large to provide isolation between many traffic classes. At the same time, the size of each buffer, S, must be large to provide good throughput for a single virtual channel. The total input buffer size is the product of these two terms T = N x S. For a router with N = 512 virtual channels and S = 4 flits, each of 576 bits, the total buffer space required is 2048 flits or 1,179,648 bits. The buffers required for the seven input channels on a router of the type described in the above patent application take more than 7Mbits of storage.

These storage requirements make it infeasible to implement single-chip fabric routers with large numbers of large virtual channel buffers in present VLSI technology which is limited to about 1Mbit per router ASIC chip. In the past this has been addressed by either having a smaller number of virtual channels, which can lead to buffer interference between different traffic classes, by making each virtual channel small (often one flit in size) which leads to poor performance on a single virtual channel, or by dividing the router across several ASIC chips which increases cost and complexity.

SUMMARY OF THE INVENTION

The present invention overcomes the storage limitations of prior-art routers by providing a small pool of on-chip flit-buffers that are used as a cache and overflowing any flits that do not fit into this pool to off-chip storage. Our simulation studies show that while buffer isolation is required to guarantee traffic isolation, in practice only a tiny fraction of the buffers are typically occupied. Thus, most of the time all active virtual channels fit in the cache and the external memory is rarely accessed.

Thus, in accordance with the present invention, a router includes buffers for information units such as flits transferred through the router. The buffers include a first set of rapidly accessible buffers for the information units and a second set of buffers for the information units that are accessed more slowly than the first set.

In the preferred embodiment, the fabric router is implemented on one or more integrated circuit chips. The first set of buffers is located on the router integrated circuit chips, and the second set of buffers is located on memory chips separate from the router integrated circuit ships. The second set of buffers may hold information units for a complete set of virtual units.

In one embodiment, the first set of buffers comprises a buffer pool and a pointer array. The buffer pool is shared by virtual channels, and the array of pointers points to information units, associated with individual channels, within the buffer pool.

In another embodiment, the first set of buffers is organized as a set associative cache. Specifically, each entry in the set associative cache may contain a single information unit or it may contain the buffers and state for an entire virtual channel.

Flow control may be provided to stop the arrival of new information units while transferring information units between the first set of buffers and the second set of buffers. The flow control may be blocking or credit based.

Miss status registers may hold the information units waiting for access to the second set of buffers. An eviction buffer may hold entries staged for transfer from the first set of buffers to the second set of buffers.

Applications of the invention include a multicomputer interconnection network, a network switch or router, and a fabric router within an internet router.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing and other objects, features and advantages of the invention will be apparent from the following more particular description of preferred embodiments of the invention, as illustrated in the accompanying drawings in which like reference characters refer to the same parts throughout the different views. The drawings are not necessarily to scale, emphasis instead being placed upon illustrating the principles of the invention.

Figure 1 illustrates an internet configuration of routers to which the present invention may be applied.

Figure 2 illustrates a three-dimensional fabric forming a router of Figure 1.

Figure 3 illustrates a fabric router used in the embodiment of Figure 2.

Figure 4 illustrates a set of input buffers previously provided in the fabric router of figure 3.

Figure 5 illustrates an input buffer array in accordance with one embodiment of the invention.

Figure 6 illustrates the pointer array and buffer pool of Figure 5 in greater detail.

Figure 7 illustrates a specific entry in the pointer array of Figure 6.

Figure 8 illustrates another embodiment of the invention in which the input buffer array is organized as a set associative cache with each entry containing a single flit.

Figure 9 illustrates a third embodiment of the invention in which the input buffer array is organized as a set associative cache in which each entry contains the flit buffers and state for an entire virtual channel.

Figure 10 presents simulation channel occupancy histograms.

DETAILED DESCRIPTION OF THE INVENTION

Although the present invention is applicable to any router application, including those in multicomputers, multiprocessors, and network switches and routers, it will be described relative to a fabric router within an internet router. Such routers are presented in the above-mentioned PCT application.

As illustrated in Figure 1, the Internet is arranged as a hierarchy of networks. A typical end-user has a workstation 22 connected to a local-area network or LAN 24. To allow users on the LAN to access the rest of the internet, the LAN is connected via a router R to a regional network 26 that is maintained and operated by a Regional

Network Provider or RNP. The connection is often made through an Internet Service Provider or ISP. To access other regions, the regional network connects to the backbone network 28 at a Network Access Point (NAP). The NAPs are usually located only in major cities.

The network is made up of links and routers. In the network backbone, the links are usually fiber optic communication channels operating using the SONET (synchronous optical network) protocol. SONET links operate at a variety of data rates ranging from OC-3 (155Mb/s) to OC-192 (9.9Gb/s). These links, sometimes called trunks, move data from one point to another, often over considerable distances.

Routers connect a group of links together and perform two functions: forwarding and routing. A data packet arriving on one link of a router is forwarded by sending it out on a different link depending on its eventual destination and the state of the output links. To compute the output link for a given packet, the router participates in a routing protocol where all of the routers on the Internet exchange information about the connectivity of the network and compute routing tables based on this information.

Most prior art Internet routers are based on a common bus or a crossbar switch. Typically, a given SONET link 30 is connected to a line-interface module. This module extracts the packets from the incoming SONET stream. For each incoming packet, the line interface reads the packet header, and using this information, determines the output port (or ports) to which the packet is to be forwarded. To forward the packet, a communication path within the router is arbitrated, and the packet is transmitted to an output line interface module. The module subsequently transmits the packet on an outgoing SONET link to the next hop on the route to its destination.

The router of the above mentioned international application overcomes the bandwidth and scalability limitations of prior-art bus- and crossbar- based routers by using a multi-hop interconnection network as a router, in particular a 3-dimensional torus network illustrated in Figure 2. With this arrangement, each router in the wide-area backbone network in effect contains a small in-cabinet network. To avoid confusion we will refer to the small network internal to each router as the switching fabric and the routers and links within this network as the fabric routers and fabric links.

In the 3-dimensional torus switching fabric of nodes illustrated in Figure 2, each node N comprises a line interface module that connects to incoming and outgoing SONET internet links. Each of these line-interface nodes contains a switch-fabric router that includes fabric links to its six neighboring nodes in the torus. IP packets that arrive over one SONET link, say on node A, are examined to determine the SONET link on which they should leave the internet router, say node B, and are then forwarded from A to B via the 3-D torus switch fabric.

Typical packets forwarded through the internet range from 50 bytes to 1.5 Kbytes. For transfer through the fabric network of the internet router of the present invention, the packets are divided into segments, or flits, each of 36 bytes. At least the header included in the first flit of a packet is modified for control of data transfer through the fabric of the router. In the preferred router, the data is transferred through the fabric in accordance with a wormhole routing protocol.

Flits of a packet flow through the fabric in a virtual network comprising a set of buffers. One or more buffers for each virtual network are provided on each node in the fabric. Each buffer is sized to hold at least one flow-control digit or flit of a message. The virtual networks all share the single set of physical channels between the nodes of the real fabric network, and a fair arbitration policy is used to multiplex the use of the physical channels over the competing virtual networks.

A fabric router used to forward a packet over the switch fabric from the module associated with its input link to the module associated with its output link is illustrated in Figure 3. The router has seven input links 58 and seven output links 60. Six of the links connect to adjacent nodes in the 3-D torus network of Figure 2. The seventh input link accepts packets from the forwarding engine 50 and the seventh output link sends packets to the packet output buffer 52 in this router's line interface module. Each input link 58 is associated with an input buffer array 62 and each output link 60 is associated with an output register 64. The input buffers and output registers are connected together by a 7 x 7 crossbar switch 66. A virtual network is provided for each pair of output nodes, and each of the seven input buffer arrays 62 contains, for example, four flit buffers for each virtual network in the machine.

If a virtual channel of a fabric router destined for an output node is free when the head flit of a packet arrives for that virtual channel, the channel is assigned to that packet for the duration of the packet, that is, until the tail flit of the packet passes. However, multiple packets may be received at a router for the same virtual channel through multiple inputs. If a virtual channel is already assigned, the new head flit must wait in its flit buffer. If the channel is not assigned, but two head flits for that channel arrive together, a fair arbitration must take place. Until selected in the fair arbitration process, flits remain in the input buffer, backpressure being applied upstream.

Once assigned an output virtual channel, a flit is not enabled for transfer across a link until a signal is received from the downstream node that an input buffer at that node is available for the virtual channel.

Prior routers have used a buffer organization, illustrated in Figure 4, in which each flit buffer is assigned to a particular virtual channel and cannot be used to hold flits associated with any other virtual channel. Figure 4 shows an arriving flit, 100, and the flit buffer array for one input port of a router, 200. The buffer array 200 contains one row for each virtual channel supported by the router. Each row contains S = 2 flit buffers 204 and 205, a pointer to the first flit in the row (F) 201, a pointer to the last flit in the row (L) 202, and an empty bit (E) 203 that indicates when there are no flits in the row.

When flit 100 arrives at the input channel, it is stored in the flit buffer at a location determined by the virtual channel identifier (VCID) field of the flit 101 and the L-field of the selected row of the flit buffer. First, the VCID is used to address the buffer array 200 to select the row 210 associated with this virtual channel. The L-field for this row points to the last flit placed in the buffer. It is incremented to identify the next open flit buffer into which the arriving flit is stored. When a particular virtual channel of the input channel is selected to output a flit, the flit buffer to be read is selected in a similar manner using the VCID of the virtual channel and the F field of the corresponding row of the buffer array.

Simulations have shown that, for typical traffic, only a small fraction of the virtual channels of a particular input port are occupied at a given time. We can exploit this behavior by providing only a small pool of buffers on chip with a full array of buffers in slower, but less expensive, off-chip storage. The buffers in the pool are dynamically assigned to virtual channels so that at any instant in time, these buffers hold flits for the fraction of virtual channels that are in use. The buffer pool is, in effect, a cache for the full array of flit buffers in off-chip storage. A simple router ASIC with an inexpensive external DRAM memory can simultaneously support large numbers of virtual channels and large buffers for each virtual channel. In a preferred embodiment there are V = 2560 virtual channels, each with S = 4 flits with each flit occupying F = 576-bit flits of storage.

A first preferred embodiment of the present invention is illustrated in Figure 5. A relatively small pool of flit buffers 400 is used to hold the currently active set of flits on router chip 500. The complete flit buffer array 200 is placed in inexpensive off-chip memory and holds any flits that exceed the capacity of the pool. A pointer array 300 serves as a directory to hold the state of each virtual channel and to indicate where each flit associated with the virtual channel is located.

A more detailed view of pointer array 300 and buffer pool 400 is shown in Figure 6. The pointer array 300 contains one row per virtual channel. Each row contains three state fields and S pointer fields. In this example, S = 4. The F-field 301 indicates which of the S = 4 pointer fields corresponds to the first flit on the channel. The L-field 302 indicates which pointer field corresponds to the last flit to arrive on the channel, and the E-field 303 if set indicates that the channel is empty and no flits are present.

Each pointer field, if in use, specifies the location of the corresponding flit. If the value of the pointer field, P, is in the range [0,B-1], where B is the number of buffers in the buffer pool, then the flit is located in buffer P of buffer pool 400. On the other hand, if P = B the pointer indicates that the corresponding flit is located in the off-chip flit-buffer array. Flits in the off-chip flit buffer array are located by VCID and flit number, not by pointer.

The use of this structure is illustrated in the example of Figure 7. The figure shows the entries in pointer array 300, buffer pool 400, and off-chip buffer array 200 for a single virtual channel (VCID = 4) that contains three flits. Two of these flits, the first and the last, are in the buffer pool while the middle flit is in the off-chip buffer array. To locate a particular flit, the VCID is used to select row 310 within pointer array 300. The state fields within the selected row specify that three flits are in the collective buffer with the first flit identified by pointer 1 (F=1) and the last flit identified by pointer 3 (L=3). Pointer 2 identifies the middle flit. Pointer 1 contains the value 5, which specifies that the first flit is in location 5 of buffer pool 400. Similarly, pointer 3 specifies that the last flit is in location 3 of buffer pool 400. Pointer 2, however, contains the value nil which indicates that the middle flit resides in the off-chip buffer array 200, at row 4 column 2. The row is determined by the VCID, in this case 4, and the column by the position of the pointer, pointer 2 in this example. In a preferred embodiment, the buffer pool contains 2n-1 buffers labeled 0 to 2n-2 for some n and the value 2n-1 denotes the nil pointer.

To see the advantage of flit buffer caching, consider our example system with V = 2560 virtual channels each with S = 4 flit buffers containing F = 576-bit flits. A conventional buffer organization (Figure 4) requires V x S x F = 5,898,240 bits of storage for each input channel of the router. Using an on-chip buffer pool with P = 255 buffers in combination with an off-chip buffer array (Figure 5), on the other hand requires a P x F = 146,880-bit buffer pool and a 37 x V = 94,720-bit pointer array for a total of 241,600 bits of on-chip storage per input channel. A 5,898,240-bit off-chip buffer array is also required. This represents a factor of 24 reduction in on-chip storage requirements.

The input channel controller uses a dual-threshold buffer management algorithm to ensure an adequate supply of buffers in the pool. Whenever the number of free buffers in the pool falls below a threshold, e.g., 32, the channel controller begins evicting flits from the buffer pool 400 to the off-chip buffer array 200 and updating pointer array 300 to indicate the change. Once flit eviction begins, it continues until the number of free buffers in the pool exceeds a second threshold, e.g., 64. During eviction, the flits to be evicted can be selected using any algorithm: random, first available, or least recently used. While an LRU algorithm gives slightly higher performance, the eviction event is so rare that the simplest possible algorithm suffices. The eviction process is necessary to keep the upstream controller from waiting indefinitely for a flit to depart from the buffer. Without eviction, this waiting would create dependencies between unrelated virtual channels, possibly leading to tree saturation or deadlock.

Prior routers, such as the one described in the above mentioned pending PCT application, employ credit-based flow control where the upstream controller keeps a credit count 73 (Figure 3), a count of the number of empty flit buffers, for each downstream virtual channel. The output-controller only forwards a flit for a particular virtual channel when it has a non-zero credit count for that VC, indicating that there is a flit-buffer available downstream. Whenever the upstream controller forwards a flit, it decrements the credit count for the corresponding VC. When the downstream controller empties a flit-buffer, it transmits a credit upstream. Upon receipt of each credit, the upstream controller increments the credit count for the corresponding VC.

With flit-buffer caching, the upstream output channel controller flow-control strategy must be modified to avoid oversubscribing this bandwidth in the unlikely event of a buffer-pool overflow. This additional flow control is needed because the off-chip buffer array has much lower bandwidth than the on-chip buffer pool and than the channel itself. Once the pool becomes full, flits must be evicted to the off-chip buffer array and transfers from all VCs must be blocked until space is available in the pool. To handle flit-buffer caching, the prior credit-based flow control mechanism is augmented by adding a buffer-pool credit count 75 that reflects the number of empty buffers in the downstream buffer pool. The upstream controller must have both a non-zero credit count 73 for the virtual channel and a non-zero buffer-pool credit count 75 before it can forward a flit downstream. This ensures that there is space in the buffer pool for all arriving flits. Initially, maximum counts are set for all virtual channels and for the buffer pool which is shared by all VCs. Each time the upstream controller forwards a flit it decrements both the credit count for the corresponding VC and the shared buffer-pool credit count. When the downstream controller sends a credit upstream for any VC, it sets a pool bit in the credit if the flit being credited was sent from the buffer pool. When the upstream controller receives a credit with the pool bit set, it increments the buffer-pool credit count as well as the VC credit count. With eviction of flits from the buffer pool to the off-chip buffer array, special pool-only credit is sent to the upstream controller to update its credit-count to reflect the change. Thus, transfer of new flits is stopped only while transferring flits between the buffer pool and the off-chip buffer array.

Alternatively, blocking flow control can be employed to prevent pool overrun rather than credit-based flow control. With this approach, a block bit in all upstream credits is set when the number of empty buffers in the buffer pool falls below a threshold. When this bit is set, the upstream output controller is inhibited from sending any flits downstream. Once the number of empty buffers is increased over the threshold, the block bit is cleared and the upstream controller may resume sending flits. Blocking flow control is advantageous because it does not require a pool credit counter in the upstream controller and because it can be used to inhibit flit transmission for other reasons.

An alternative preferred embodiment dispenses with the pointer array and instead uses a set-associative cache organization as illustrated in Figure 8. This organization is comprised of a state array 500, one or more cache arrays 600, and an eviction FIFO 800. An off-chip buffer array 200 (not shown) is also employed to back up the cache arrays. A flit associated with a particular buffer, B, of a particular virtual channel, V, is mapped to a possible location in each of the cache arrays in a manner similar to a conventional set-associative cache (see, for example Hennessey and Patterson, Computer Architecture: A Quantitative Approach, Second Edition, Morgan

Kaufmann, 1996, Chapter 5). The flit stored at each location is then recorded in an associated cache tag. A valid bit in each cache location signals if the entry it contains represents valid data.

The allowed location for a particular flit F={V:B} in the cache is determined by the low order bits of F. For example, consider a case with V = 2560 virtual channels, S= 4 buffers per virtual channel, and C = 128 entries in each of A = 2 cache arrays. In this case, the flit identifier F is 14-bits, 12-bits of VCID, and 2-bits of buffer identifier, B. The seven-bit cache array index is constructed by appending B to the low-order 5-bits of V, I = {V[4:0]:B}. The remaining seven high-order bits of V are then used for the cache tag, T = V[11:5].

When a flit arrives over the channel, its VCID is used to index the state array 500 to read the L field. This field is then incremented to determine the buffer location, B, within the virtual channel, into which the flit is to be stored. The array index, I, is then formed by concatenating B with the low 5-bits of the VCID and this index is used to access the cache arrays 600. While two cache arrays are shown, it is understood that any number of arrays may be employed. One of the cache arrays is then selected to receive this flit using a selection algorithm. Preference may be given to an array that contains an invalid entry in this location, or, if all entries are valid, the array that contains the location that has been least recently used. If the selected cache array already contains a valid flit, this flit is first read out into the eviction FIFO 800 along with its identity (VCID and B). The arriving flit is then written into the vacated location, the tag on that location is updated with the high-bits of the VCID, and the location is marked valid.

When a request comes to read the next flit from a particular virtual channel, the VCID is again used to index the state array 500, and the F field is read to determine the buffer location, B, to be read. The F field is then incremented and written back to the state array. The VCID and buffer number, B, are then used to search for the flit in three locations. First, the index is formed as above I = {V[4:0],B} and the cache arrays are accessed. The tag accessed from each array is compared to V[11:5] using comparator

701. If there is a match and the valid bit is set, then the flit has been located and is read out of the corresponding array via tri-state buffer 702. The valid bit of the entry containing the flit is then cleared to free this entry for later use.

If the requested flit is not found in any of the cache arrays, the eviction FIFO is then searched to determine if it contains a valid entry with matching VCID and B. If a match is found, the flit is read out of the eviction FIFO and the valid bit of the entry is cleared to free the location. Finally, if the flit is not found in either the cache arrays or the eviction FIFO, off-chip flit array 200 is read at location {V[11:0],B} to retrieve the flit from backing store.

As with the first preferred embodiment, flow control must be employed and eviction performed to make sure that requests from the upstream controller do not overrun the eviction FIFO. Either credit-based or blocking flow control can be employed as described above to prevent the upstream controller from sending flits when free space in the eviction FIFO falls below a threshold. Flits from the eviction FIFO are also written back to the off-chip flit array 200 when this threshold is exceeded.

Compared to the first preferred embodiment, the set associative organization requires fewer on-chip memory bits to implement but is likely to have a lower hit rate due to conflict misses. For the example numbers above, the state array requires 2560 x 5 = 12,800 bits of storage, and cache arrays with 256 flit-sized entries requires 256 x (576 + 1 + 7) = 149,504 bits, and an eviction FIFO with 16 entries requires 16 x (576 + 14) = 9,440 bits for a total of 171,744 bits compared to 241,600 bits for the first preferred embodiment.

Conflict misses occur because a flit cannot reside in any flit buffer, but rather only in a single location of each array (a set of buffers). Thus, an active flit may be evicted from the array before all buffers are full because several other flits map to the same location. However, the effect of these conflict misses are mitigated somewhat by the associative search of the eviction FIFO which acts as a victim cache (see Jouppi, "Improving Direct Mapped Cache Performance by the Addition of a Small Fully-

Associative Cache and Prefetch Buffers," Proceedings of 17th Annual International Symposium on Computer Architecture, 1990, pp 364-375.

The storage requirements of a flit-cache can be reduced further using a third preferred embodiment illustrated in Figure 9. This embodiment also employs set-associative cache organization. However, unlike the embodiment of Figure 8, it places all of the state and buffers associated with a given virtual channel into a single cache entry of array 900. While a single cache array 900 is shown (a direct-mapped organization), one skilled in the art will understand that any number of cache arrays may be employed. Placing all of the state in the cache arrays eliminates the need for state array 500. The eviction FIFO 1000 for this organization also contains all state and buffers for a virtual channel (but does not need a B field). Similarly, the off-chip flit array 200 (not shown) is augmented to include the state fields (F,L, and E) for each virtual channel.

When a flit arrives at the buffer, the entry for the flit's virtual channel is located and brought on chip if it is not already there. The entry is then updated to insert the new flit and update the L field. Specifically, the VCID field of the arriving flit is used to search for the virtual channel entry in three locations. First, the cache arrays are searched by using the low bits of the VCID, e.g., V[6:0], as an index and the high bits of the VCID, e.g., V[11:7] as a tag. If the stored tag matches the presented tag and the valid bit is set, the entry has been found. In this case, the L field of the matching entry is read and used to select the location within the entry to store the arriving flit. The L entry is then incremented and written back to the entry.

In the case of a cache miss, no match in the cache arrays, one of the cache arrays is selected to receive the required entry as described above. If the selected entry is currently valid holding a different entry, that entry is evicted to the eviction FIFO 1000. The eviction FIFO is then searched for the required entry. If found, it is loaded from the buffer into the selected cache array and updated as described above. If the entry is not found in the eviction FIFO, it is fetched from the off-chip buffer array. To allow other arriving flits to be processed while waiting for an entry to be loaded from off chip, the pending flit is temporarily stored in a miss holding register (1002) until the off-chip reference is complete. Once the entry has been loaded from off-chip, the update proceeds as described above.

To read a flit out of the cache given a VCID, the search proceeds in a manner similar to the write. The virtual channel entry is loaded into the cache, from the eviction FIFO or off-chip buffer array, if it is not already there. The F field of the entry is used to select the flit within the entry for readout. Finally, the F field is incremented and written back to the entry.

As with the other preferred embodiments, flow control, either blocking or credit-based is required to stop the upstream controller from sending flits when the number of empty locations in either the eviction FIFO or the miss holding registers fall below a threshold value.

The advantage of the third preferred embodiment is the small size of its on-chip storage arrays when there are very large numbers of virtual channels. Because there are no per-virtual-channel on-chip arrays, the size is largely independent of the number of virtual channels (the size of the tag field does increase logarithmically with the number of virtual channels). For example for our example parameters, V=2560, S=4, and F=576, an on-chip array with 64 entries (256 flits) contains 64 (((S (F) + 12) = 148,224 bits. An eviction FIFO and miss holding register array with 16 entries each add an additional 37,152 and 9,328 bits respectively. The total amount of on-chip storage, 194,704 is slightly larger than the 171,744 for the second preferred embodiment, but remains essentially constant as the number of virtual channels is increased beyond 2560.

Figure 10 illustrates the effectiveness of flit buffer caching employing any of the three preferred embodiments. The figure displays the results of simulating a 512-node 8x8x8 3-dimensional torus network with traffic to each destination run on a separate virtual channel. During this simulation, the occupancy of virtual channels was recorded at each point in time. The figure shows two histograms of this channel occupancy corresponding to the network operating at 30% of its maximum capacity, a typical load, and at 70% of its maximum capacity, an extremely heavy load. Even at 70% of capacity, the probability is less than 10^{-5} that more than 38 virtual channel buffers will be occupied at a given point in time. This suggests that a flit buffer cache with a capacity of 38xS flits should have a hit ratio of more than 99.999% (or conversely a miss ratio of less than 0.001%). If we extrapolate this result, it suggests that a flit buffer cache with a 256-flit capacity will have a vanishingly small miss ratio.

While we have described particular arrangements of the flit cache for the three preferred embodiments, one skilled in the art of fabric router design will understand that many alternative arrangements and organizations are possible. For example, while we have described pointer-based and set-associative organizations, one could also employ a fully-associative organization (particularly for small cache sizes), a hash table, or a tree-structured cache. While we have described cache block sizes of one flit and one virtual channel, other sizes are possible. Also, while we have described a particular encoding of virtual-channel state with the fields F, L, and E, many other encodings are possible. Moreover, while we have cached only the contents of flits and the input virtual channel state, the caching could be extended to cache the output port associated with a virtual channel and the output virtual channel state.

While this invention has been particularly shown and described with references to preferred embodiments thereof, it will be understood by those skilled in the art that various changes in form and details may be made therein without departing from the spirit and scope of the invention as defined by the appended claims. For example, though the preferred embodiments provide flit buffers in fabric routers, the invention can be extended to other information units, such as packets and messages, in other routers.

CLAIMS

What is claimed is:

1. A router including buffers, for information units transferred through the router, comprising:

    a first set of rapidly accessible buffers for the information units; and

    a second set of buffers for the information units that are accessed more slowly than the first set.

2. A router as claimed in claim 1 wherein:

    the router is implemented on one or more integrated circuit chips;

    the first set of buffers is located on the router integrated circuit chips; and

    the second set of buffers is located on memory chips separate from the router integrated circuit chips.

3. A router as claimed in claim 1 where the second set of buffers holds information units for a complete set of virtual channels.

4. A router as claimed in claim 1 wherein the first set of buffers comprises:

    a buffer pool; and

    a pointer array.

5. A router as claimed in claim 1 wherein the first set of buffers is organized as a set-associative cache.

6. A router as claimed in claim 5 wherein each entry in the set associative cache contains a single information unit.

7. A router as claimed in claim 5 wherein each entry in the set associative cache contains the buffers and state for an entire virtual channel.

8. A router as claimed in claim 1 further comprising flow control to stop the arrival of new information units while transferring information units between the first set of buffers and the second set of buffers.

9. A router as claimed in claim 8 wherein the flow control is blocking.

10. A router as claimed in claim 8 wherein the flow control is credit-based.

11. A router as claimed in claim 1 further comprising miss status registers to hold information units waiting for access to the second set of buffers.

12. A router as claimed in claim 1 further comprising an eviction buffer to hold entries staged for transfer from the first set of buffers to the second set of buffers.

13. A router as claimed in claim 1 in a multicomputer interconnection network.

14. A router as claimed in claim 1 in a network switch or router.

15. A router as claimed in claim 1 wherein the router is a fabric router and the information units are flits.

16. A method of buffering information units in a router comprising:

storing the information units in a first set of rapidly accessible buffers; and storing overflow from the first set of buffers in a second set of buffers that are accessed more slowly than the first set.

17. A method as claimed in claim 16 wherein

the router is implemented on one or more integrated circuit chips;

the first set of buffers are located on the router integrated circuit chips; and

the second set of buffers are located on memory chips separate from the router integrated circuit chips.

18. A method as claimed in claim 16 where the second set of buffers holds information units for a complete set of virtual channels.

19. A method as claimed in claim 16 further comprising, in the first set of buffers, storing the information units in a buffer pool shared by channels and pointing to information units within the buffer pool from an array of pointers associated with individual channels.

20. A method as claimed in claim 16 wherein the first set of buffers is organized as a set-associative cache.

21. A method as claimed in claim 20 wherein each entry in the set associative cache contains a single information unit.

22. A method as claimed in claim 20 wherein each entry in the set associative cache contains the information unit buffers and state for an entire virtual channel.

23. A method as claimed in claim 16 further comprising controlling flow to stop the arrival of new information units while transferring flits between the first set of buffers and the second set of buffers.

24. A method as claimed in claim 23 wherein the flow control is blocking.

25. A method as claimed in claim 23 wherein the flow control is credit-based.

26. A method as claimed in claim 16 further comprising storing information units waiting for access to the second set of flit buffers in miss status registers.

27. A method as claimed in claim 16 further comprising storing information units staged for transfer from the first set of buffers to the second set of buffers in an eviction buffer.

28. A method as claimed in claim 16 wherein the router is in a multicomputer interconnection network.

29. A method as claimed in claim 16 wherein the fabric router is in a network switch or router.

30. A method as claimed in claim 16 wherein the router is a fabric router and the information units are flits.

31. A network comprising a plurality of interconnected routers, each router including information unit buffers comprising:

a first set of rapidly accessible information unit buffers; and

a second set of information unit buffers that are accessed more slowly than the first set.

32. A network as claimed in claim 31 wherein:

the router is implemented on one or more integrated circuit chips;

the first set of buffers are located on the router integrated circuit chips; and

the second set of buffers are located on memory chips separate from the router integrated circuit chips.

33. A network as claimed in claim 31 where the second set of buffers holds information units for a complete set of virtual channels.

34. A network as claimed in claim 31 wherein the first set of buffers comprises:

a buffer pool; and

a pointer array.

35. A network as claimed in claim 31 wherein the first set of buffers is organized as a set-associative cache.

36. A network as claimed in claim 31 further comprising flow control to stop the arrival of new information units while transferring information units between the first set of buffers and the second set of buffers.

37. A network as claimed in claim 31 further comprising flow control to stop the arrival of new information units while transferring flits between the first set of buffers and the second set of buffers.

38. A network as claimed in claim 31 in a network switch or router.

39. A network as claimed in claim 31 wherein the router is a fabric router and the information units are flits.

40. A router comprising:

means for storing information units in a first set of rapidly accessible buffers; and

means for storing information units in a second set of buffers that are accessed more slowly than the first set.

41. A router as claimed in claim 40 where the second set of buffers holds information units for a complete set of virtual channels.

42. A router as claimed in claim 40 wherein the first set of buffers comprises:

a buffer pool shared by channels; and

means for pointing to entries in the buffer pool for individual channels.

43. A router as claimed in claim 40 wherein the first set of buffers is organized as a set-associative cache.

44. A router as claimed in claim 40 further comprising means for providing flow control for stopping the arrival of new information units while transferring information units between the first set of buffers and the second set of buffers.

45. A router as claimed in claim 40 wherein the router is a fabric router and the information units are flits.

FABRIC ROUTER WITH FLIT CACHING

ABSTRACT OF THE DISCLOSURE

In a fabric router, flits are stored on chip in a first set of rapidly accessible flit buffers, and overflow from the first set of flit buffers is stored in a second set of off-chip flit buffers that are accessed more slowly than the first set. The flit buffers may include a buffer pool accessed through a pointer array or a set associative cache. Flow control between network nodes stops the arrival of new flits while transferring flits between the first set of buffers and the second set of buffers.

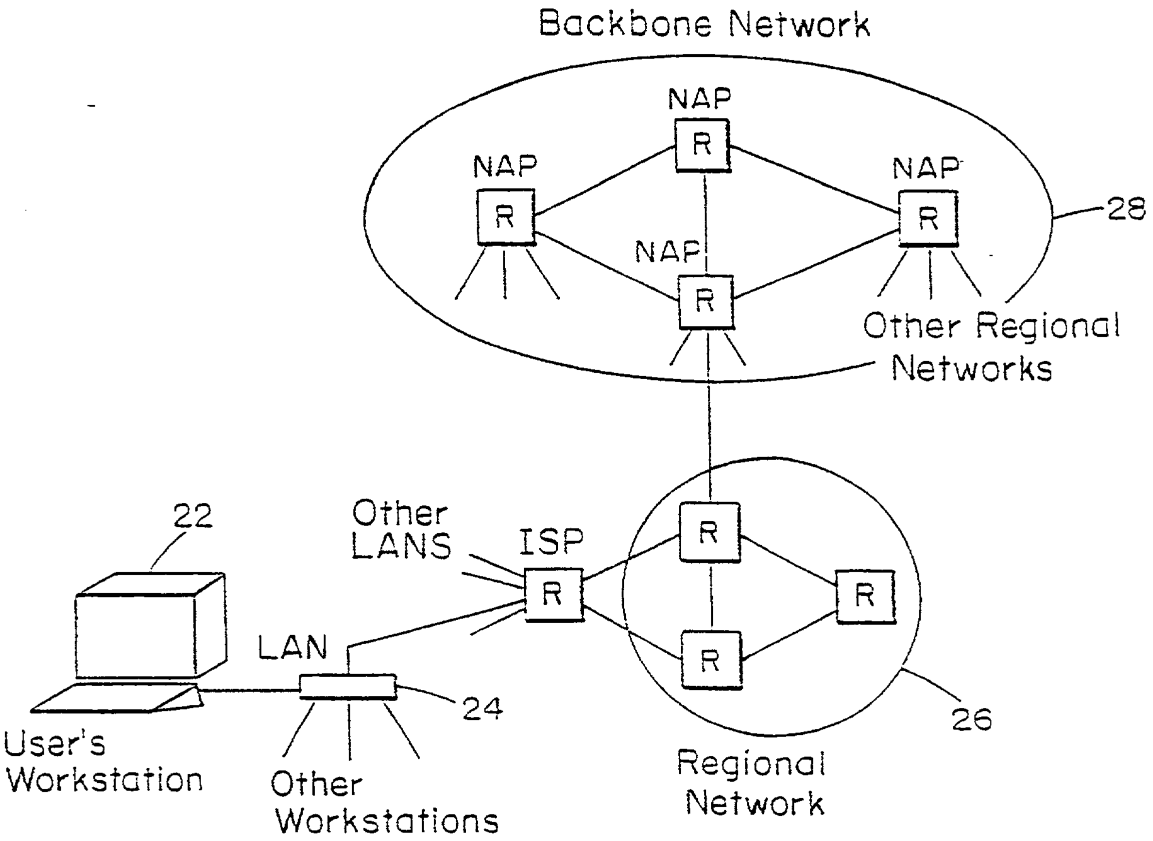

FIG. 1
PRIOR ART

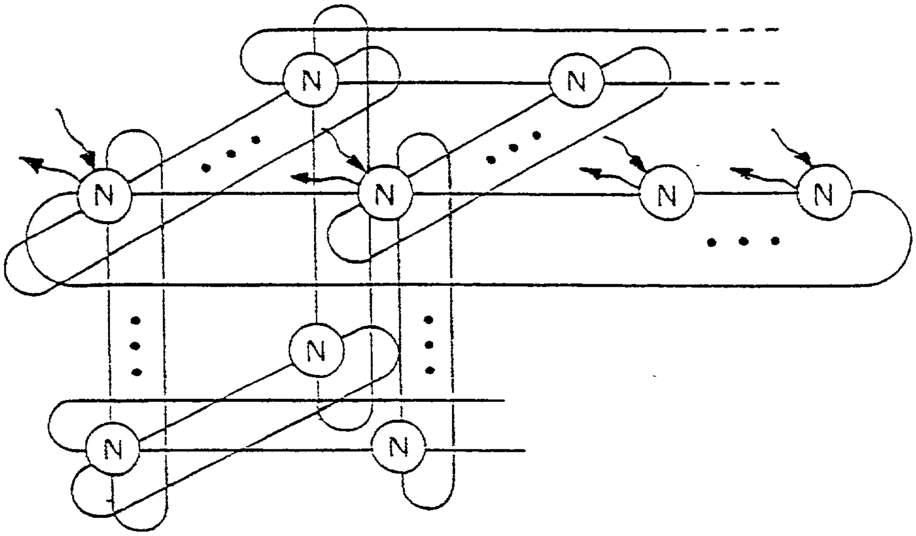

FIG. 2

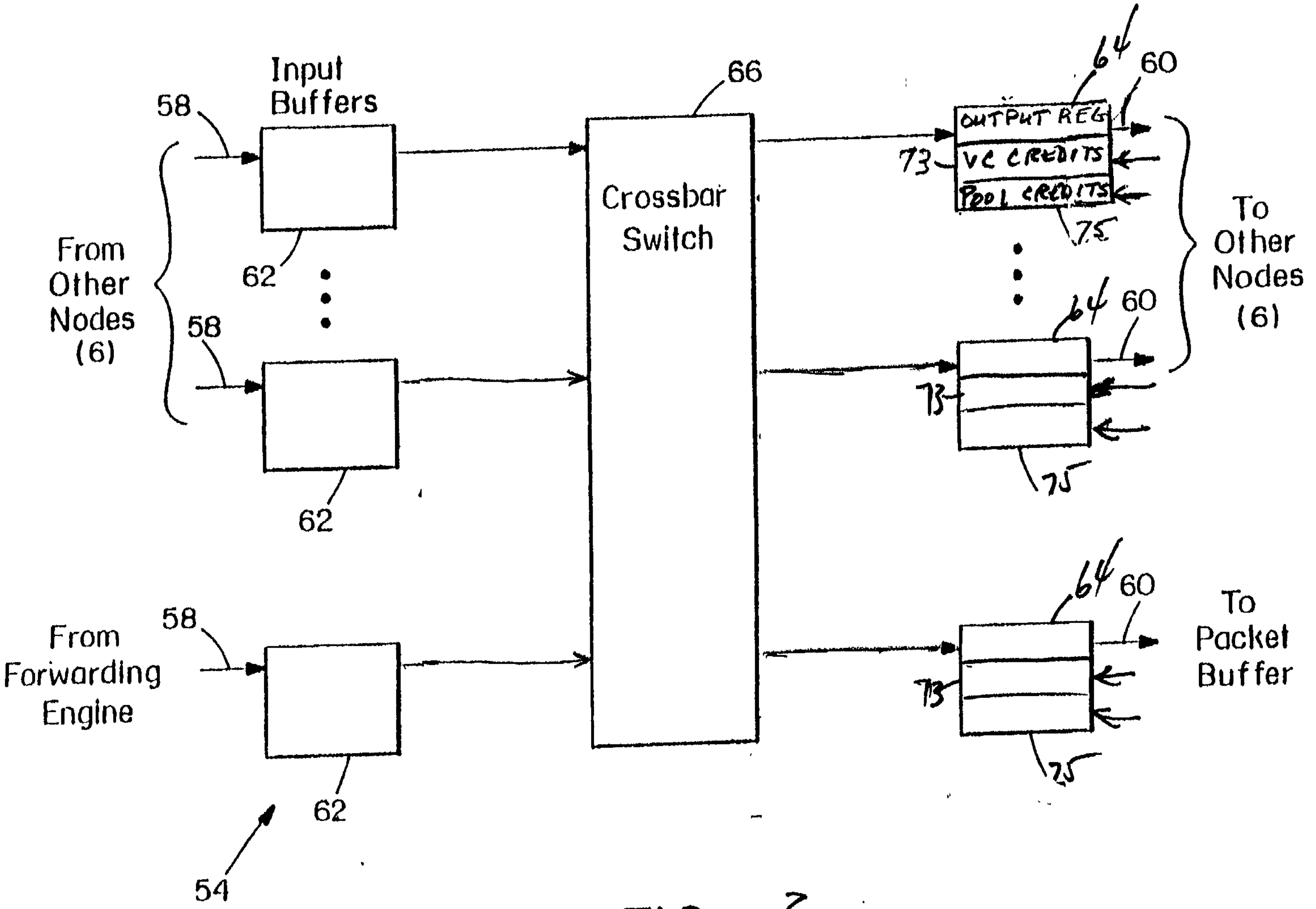
Input
Buffers
58
62
58
62
From
Other
Nodes
(6)
58
From
Forwarding
Engine
62
54
66
Crossbar
Switch
64
60
OUTPUT REG
VC CREDITS
POOL CREDITS
73
75
64
60
73
75
To
Other
Nodes
(6)
64
60
73
75
To
Packet
Buffer

FIG. 3

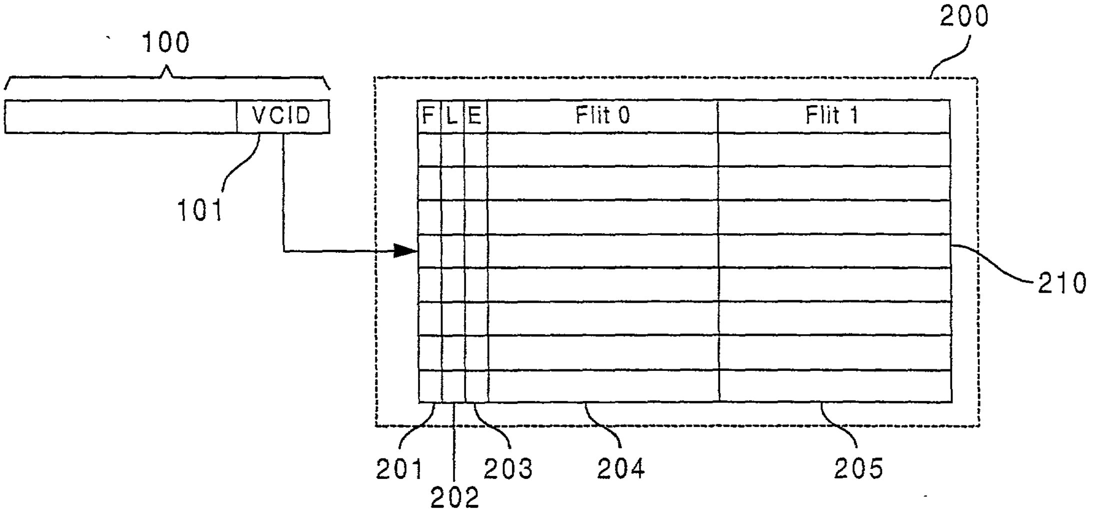

Figure 4

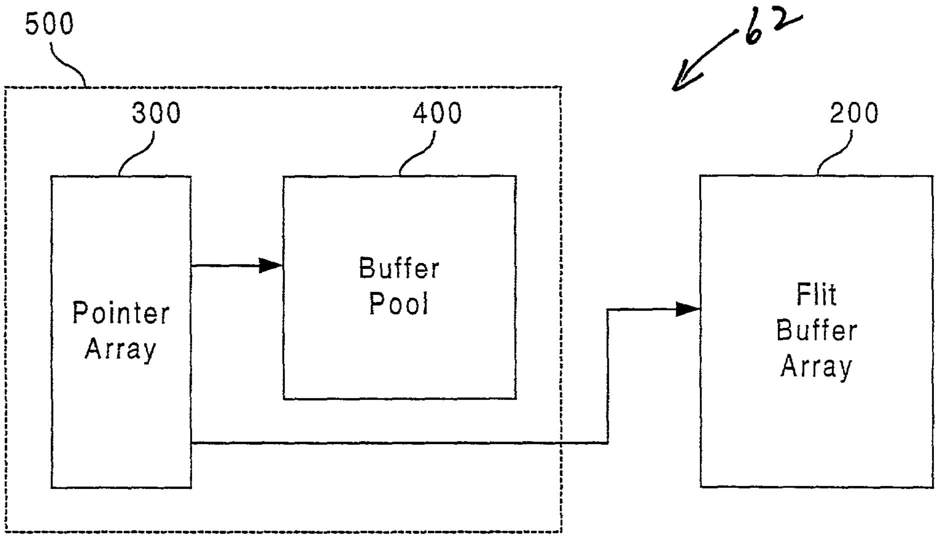

Figure 5

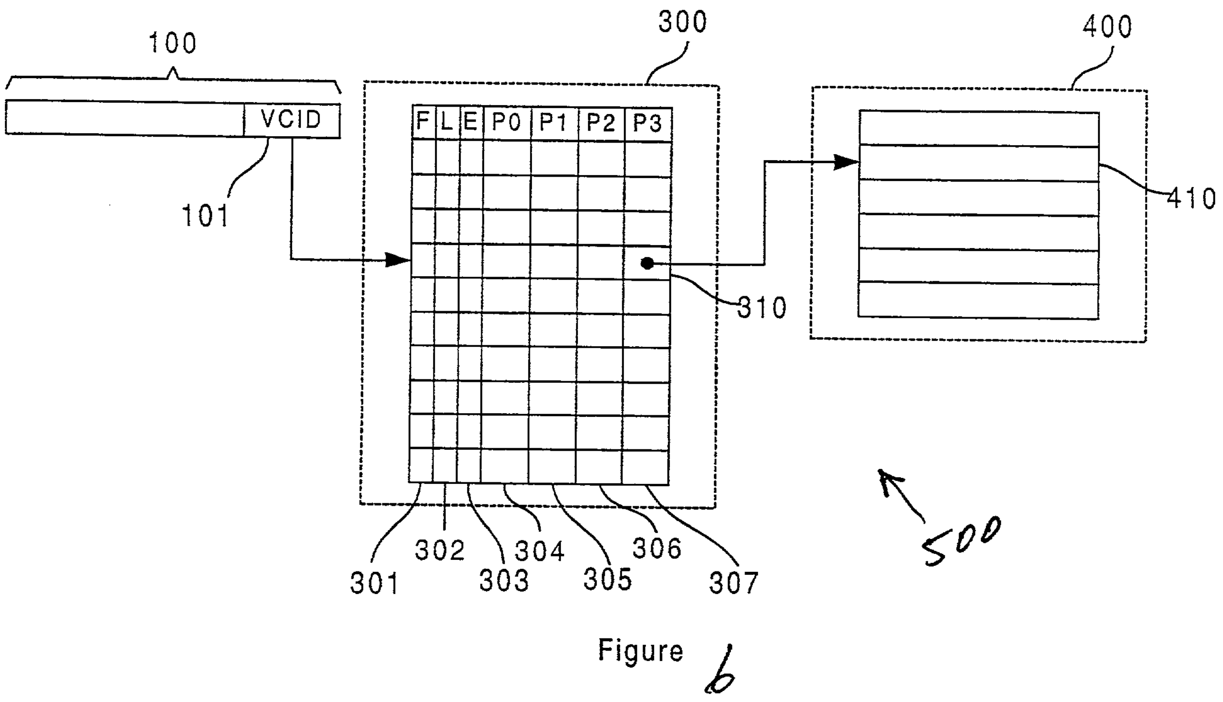

Figure 6

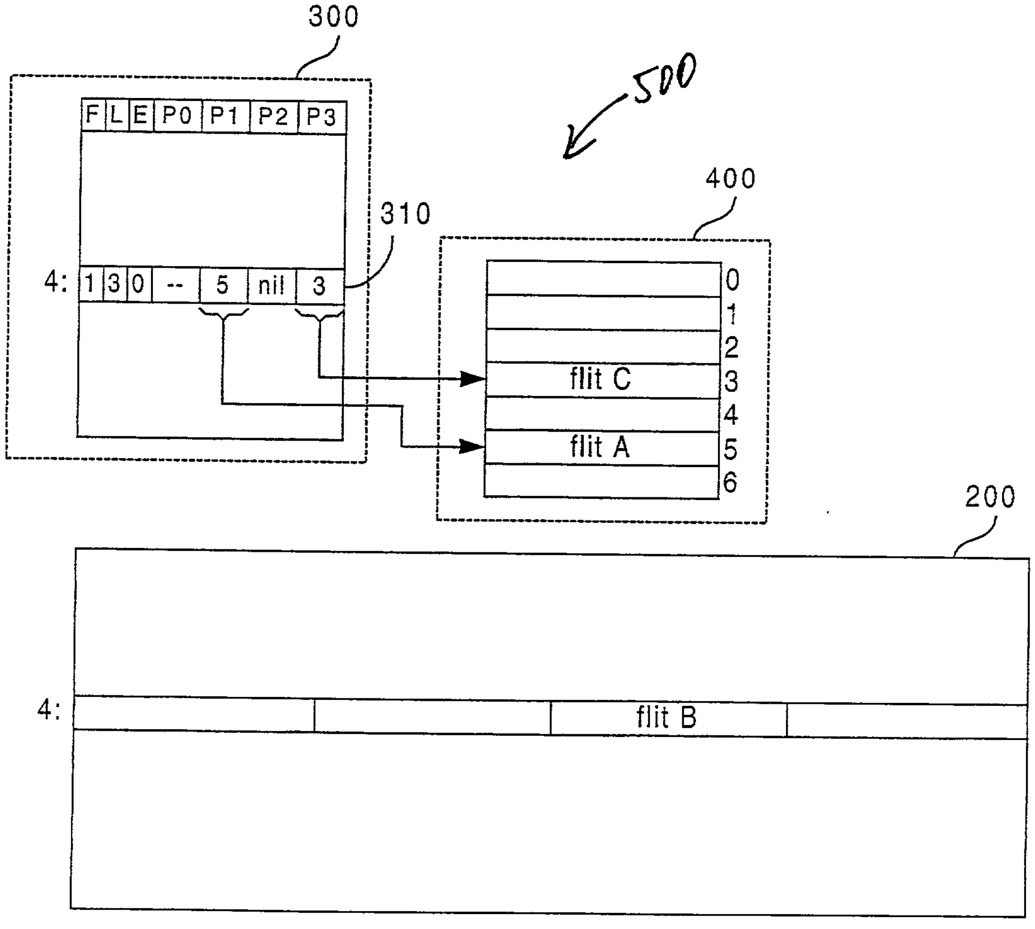

Figure 7

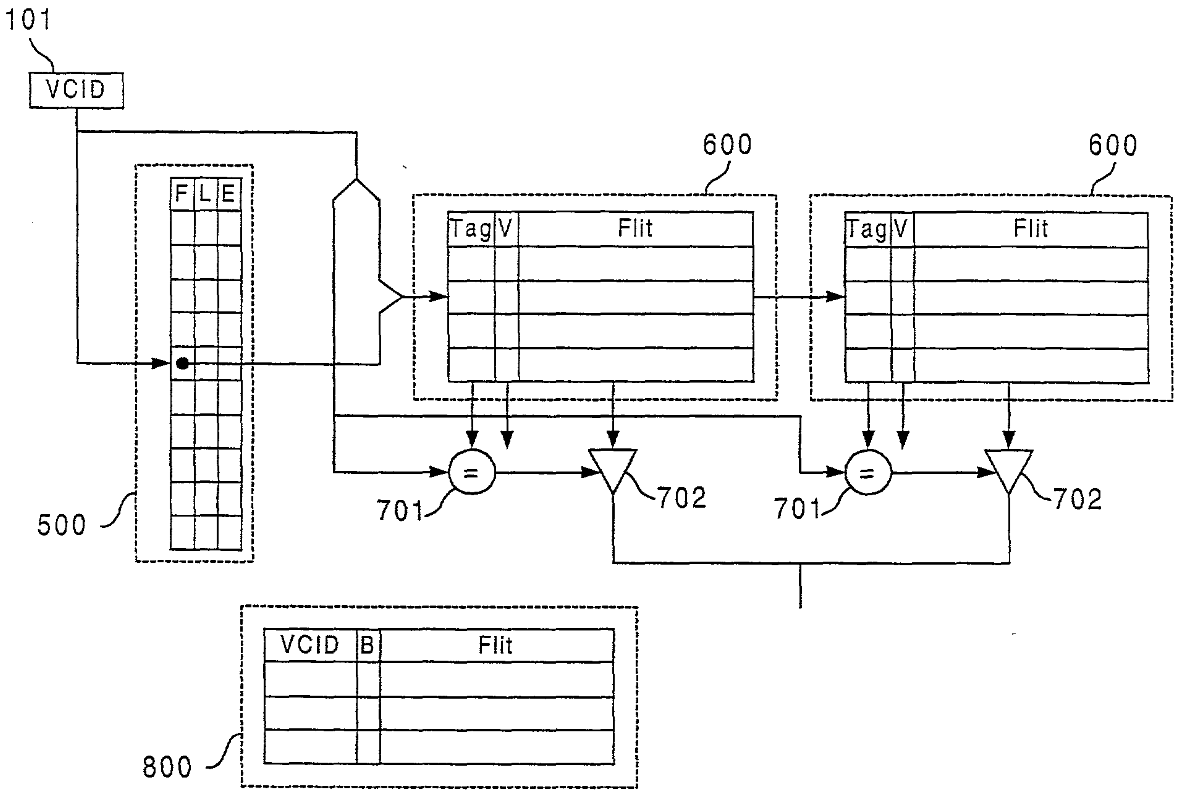

Figure 8

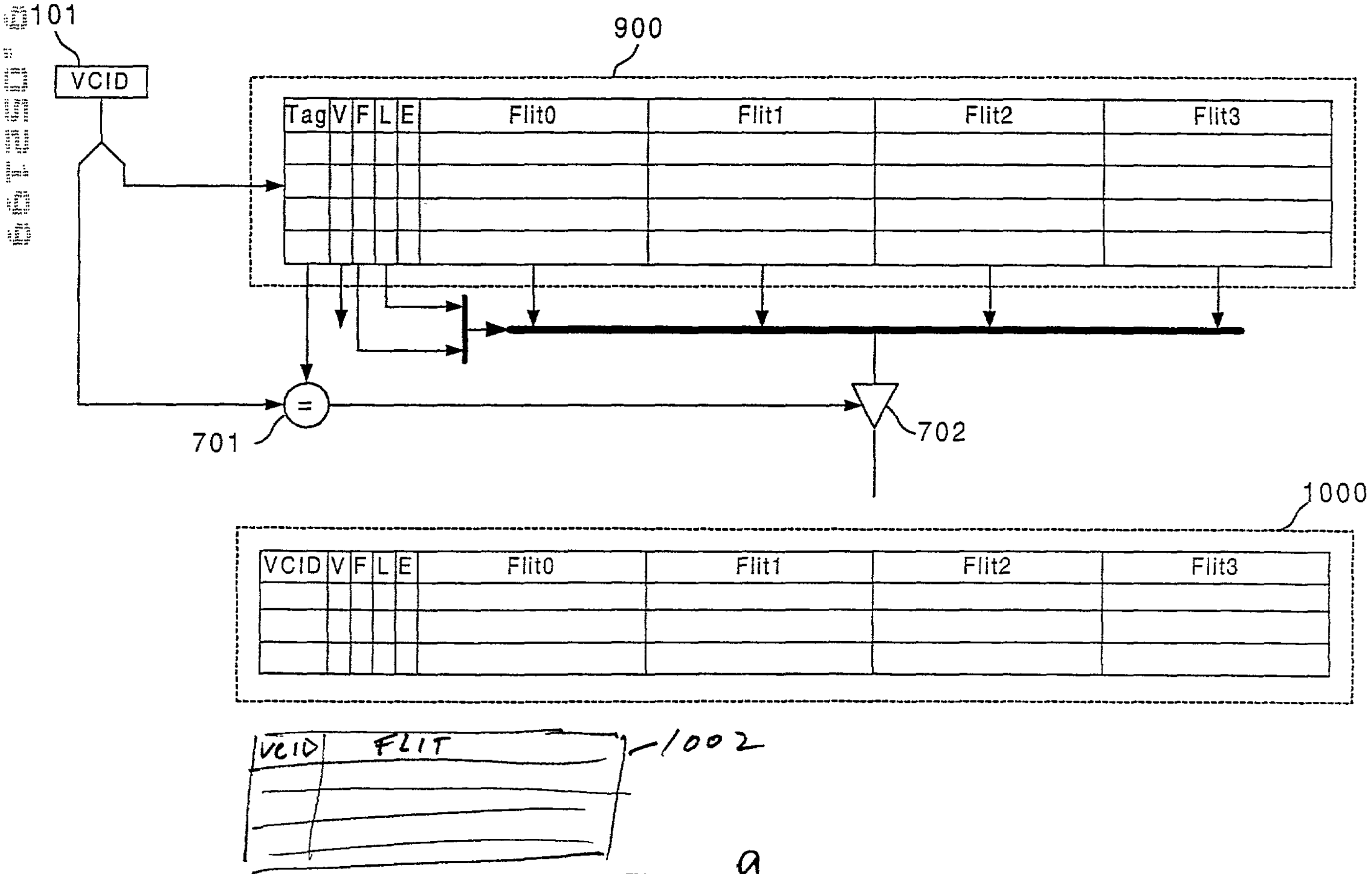
101
VCID
900
Tag V F L E Flit0 Flit1 Flit2 Flit3
701
702
1000
VCID V F L E Flit0 Flit1 Flit2 Flit3
VCID FLIT
1002

Figure 9

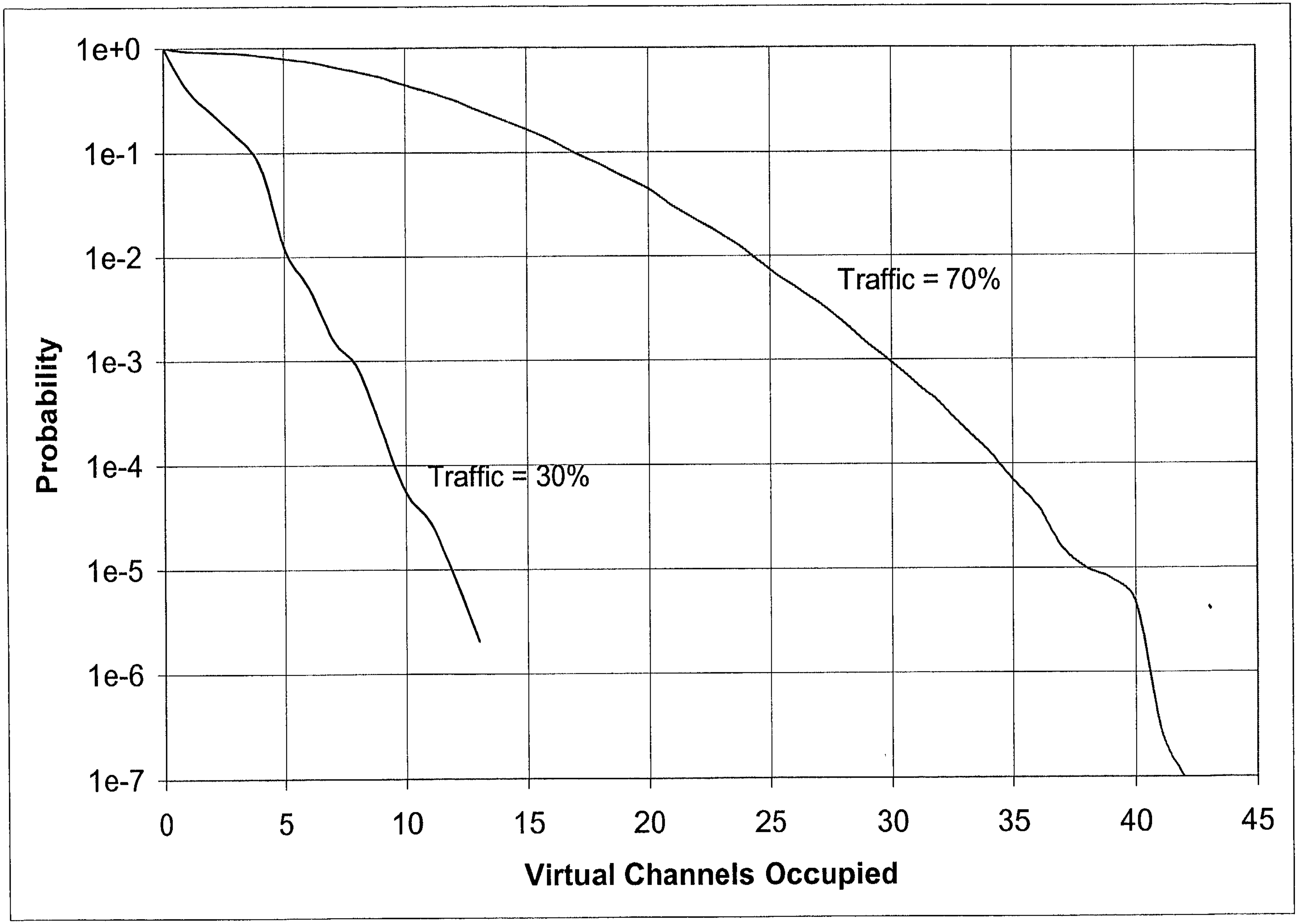
1e+0
1e-1
1e-2
1e-3
1e-4
1e-5
1e-6
1e-7
Probability
0
5
10
15
20
25
30
35
40
45
Virtual Channels Occupied
Traffic = 70%
Traffic = 30%

FIG. 10